Similarity-Guided Clause Generalization

S. Ferilli, T.M.A. Basile, N. Di Mauro, M. Biba, and F. Esposito

Dipartimento di Informatica
Università di Bari
via E. Orabona, 4 - 70125 Bari - Italia
{ferilli, basile, ndm, biba, esposito}@di.uniba.it

Abstract. Few works are available in the literature to define similarity criteria between First-Order Logic formulæ, where the presence of relations causes various portions of one description to be possibly mapped in different ways onto another description, which poses serious computational problems. Hence, the need for a set of general criteria that are able to support the comparison between formulæ. This could have many applications; this paper tackles the case of two descriptions (e.g., a definition and an observation) to be generalized, where the similarity criteria could help in focussing on the subparts of the descriptions that are more similar and hence more likely to correspond to each other, based only on their syntactic structure. Experiments on real-world datasets prove the effectiveness of the proposal, and the efficiency of the corresponding implementation in a generalization procedure.

1 Introduction

First-order logic (*FOL* for short) is a powerful formalism, that is able to express relations between objects and hence can overcome the limitations shown by propositional or attribute-value representations. However, the presence of relations causes various portions of one description to be possibly mapped in different ways onto another description, which poses problems of computational effort when two descriptions have to be compared to each other. Hence, the availability of techniques for the comparison between FOL (sub-)descriptions could have many applications, particularly in the Artificial Intelligence community: helping a subsumption procedure to converge quickly, assessing a degree of similarity between two formulæ, implementing a *flexible matching* procedure, supporting instance-based classification techniques or *conceptual clustering*.

As to supervised learning, many systems generalize definitions against observations, and a similarity function could help the procedure in focussing on the components that are more similar and hence more likely to correspond to each other. Clearly, this concerns the semantic aspects of the domain, and hence there is no precise (i.e., algorithmic) way for recognizing the correct (sub-)formulæ. Thus, the problem must be attacked heuristically, by developing some method that can hypothesize which part of a description refers to which part of the other, based only on their syntactic structure. To these aims, partial similarities among description components must be searched for.

Specifically, many first-order Machine Learning systems infer theories in the form of Logic Programs, a restriction of FOL to sets of *Horn clauses*, i.e. logical formulæ of the form $l_1 \wedge \cdots \wedge l_n \Rightarrow l_0$ where the l_i's are *atoms*, usually represented in Prolog style as l_0 :- $l_1, \ldots, l_n$ to be interpreted as "l_0 (called *head* of the clause) is true, provided that l_1 and ... and l_n (called *body* of the clause) are all true". Without loss of generality [8], we will deal with the case of linked Datalog clauses.

In the following sections, some criteria and a formula on which basing similarity considerations between first-order logic clauses will be presented, that are intended to represent a good tradeoff between significance, effectiveness and expressiveness on one side, and computational efficiency on the other. Then, Section 5 will show how the proposed formula and criteria, are able to effectively guide a clause generalization procedure. Lastly, Section 4 will deal with related work, while 6 will conclude the paper and outline future work directions.

2 Similarity Formula

Intuitively, the evaluation of similarity between two items i' and i'' might be based both on the presence of common features, which should concur in a positive way to the similarity evaluation, and on the features of each item that are not owned by the other (let us define this as the *residual* of the former with respect to the latter), which should concur negatively to the whole similarity value assigned to them [5]. Thus, plausible similarity parameters:

n , the number of features owned by i' but not by i'' (*residual* of i' wrt i'');
l , the number of features owned both by i' and by i'';
m , the number of features owned by i'' but not by i' (*residual* of i'' wrt i').

We developed a novel similarity function that expresses the degree of similarity between i' and i'' based on the above parameters:

$$sf(i', i'') = \mathrm{sf}(n, l, m) = \frac{l+1}{l+n+2} + \frac{l+1}{l+m+2} \qquad (1)$$

It takes values in $]0, 1[$, which resembles the theory of probability and hence can help human interpretation of resulting value. A complete overlapping of the model onto the observation tends to the limit of 1 as long as the number of common features grows. The full-similarity value 1 is never reached, which is consistent with the intuition that the only case in which this should happen is the exact identification of items, i.e. $i' = i''$ (in the following, we assume $i' \neq i''$). Conversely, in case of no overlapping the function will tend to 0 as long as the number of non-shared features grows. This is consistent with the intuition that there is no limit to the number of different features owned by the two descriptions, which contribute to make them ever different. Moreover, in case of no features at all in two descriptions ($n = l = m = 0$, e.g., two objects with no characteristics associated) the function evaluates to $1/2$, which can be considered intuitively correct for a case of maximum uncertainty. For instance, one such case is when a model includes an object for which there are no properties to be fulfilled: when comparing it to an observed object without properties as well, one cannot know if the overlapping is actually total because in fact both descriptions have no property at all to be fulfilled, or it just happens that previous generalizations have dropped from the model all the features that it previously owned. Note that each of the two terms refers specifically to one of the two clauses under comparison, and hence a weight could be introduced to give different importance to either of the two (this might be typically needed when the comparison concerns a model against an observation); this is, however, outside the scope of this paper.

3 Similarity Criteria

In FOL formulæ, terms represent specific objects; unary predicates generally represent term properties and n-ary predicates express relationships. Hence, two levels of similarity can be defined for pairs of first-order descriptions: the *object* level, concerning similarities between terms in the descriptions, and the *structure* one, referring to how the nets of relationships in the descriptions overlap.

Example 1. Let us consider, as a running example throughout the paper, the following two clauses (in this case, a rule C and a classified observation E):

$C : h(X)$:- $p(X,Y), p(X,Z), p(W,X), r(Y,U), o(Y,Z), q(W,W), s(U,V),$
$\pi(X), \phi(X), \rho(X), \pi(Y), \sigma(Y), \tau(Y), \phi(Z), \sigma(W), \tau(W), \pi(U), \phi(U).$

$E : h(a)$:- $p(a,b), p(a,c), p(d,a), r(b,f), o(b,c), q(d,e), t(f,g),$
$\pi(a), \phi(a), \sigma(a), \tau(a), \sigma(b), \tau(b), \phi(b), \tau(d), \rho(d), \pi(f), \phi(f), \sigma(f).$

3.1 Object Similarity

Consider two clauses C' and C''. Call $A' = \{a'_1, \ldots, a'_n\}$ the set of terms in C', and $A'' = \{a''_1, \ldots, a''_m\}$ the set of terms in C''. When comparing a pair $(a', a'') \in A' \times A''$, i.e. an object taken from C' and one taken from C'', respectively, two kinds of object features can be distinguished: the properties they own as expressed by unary predicates (*characteristic features*), and the ways in which they relate to other objects according to n-ary predicates (*relational features*). More precisely, relational features are defined by the position the object holds among the n-ary predicate arguments, since different positions actually refer to different roles played by the objects. In the following, we will refer to a *role* as a couple $R = (predicate, position)$ (written compactly as $R = predicate/arity.position$). For instance, characteristic features could be `male(X)` or `tall(X)`, while relational features could be expressed by predicates such as `parent(X,Y)`, where specifically the first argument position identifies the 'parent' role (*parent*/2.1), and the second one represents the 'child' role (*parent*/2.2).

Two corresponding similarity values can be associated to a' and a'': a *characteristic similarity*, where (1) is applied to values related to the characteristic features, and a *relational similarity*, based on how many times the two objects play the same or different roles in the n-ary predicates.

The characteristic similarity between a' and a'', $\mathrm{sf}_c(a', a'')$, can be computed, by considering the set P' of properties related to a' and the set P'' of properties related to a'', as $\mathrm{sf}(n_c, l_c, m_c)$ for the following parameters:

$n_c = |P' \setminus P''|$ is the number of properties owned by the object represented by term a' in C' but not by the object represented by term a'' in C'' (*characteristic residual* of a' wrt a'');

$l_c = |P' \cap P''|$ is the number of common properties between the object represented by term a' in C' and the object represented by term a'' in C'';

$m_c = |P'' \setminus P'|$ is the number of properties owned by the object represented by term a'' in C'' but not by the object represented by term a' in C' (*characteristic residual* of a'' wrt a').

A similar technique can be applied to compute the relational similarity between a' and a''. In this case, due to the possibility that one object plays multiple times the same role in different relations (e.g., a parent of many children), we have to consider the *multi*sets R' and R'' of roles played by a' and a'', respectively. Hence, the relational similarity between a' and a'', $\mathrm{sf}_r(a', a'')$, can be computed as $\mathrm{sf}(n_r, l_r, m_r)$ for the following parameters:

$n_r = |R' \setminus R''|$ expresses how many times a' plays in C' role(s) that a'' does not play in C'' (*relational residual* of a' wrt a'');

$l_r = |R' \cap R''|$ is the number of times that both a' in C' and a'' in C'' play the same role(s);

$m_r = |R'' \setminus R'|$ expresses how many times a'' plays in C'' role(s) that a' does not play in C' (*relational residual* of a'' wrt a').

Overall, we can define the *object similarity* between two terms as $\mathrm{sf}_o(a', a'') = \mathrm{sf}_c(a', a'') + \mathrm{sf}_r(a', a'')$.

Example 2. The properties and roles for some terms in C and E, and the comparison for some of the possible pairs, are reported in Table 1.

3.2 Structural Similarity

When checking for the structural similarity of two formulæ, many objects can be involved, and hence their mutual relationships represent a constraint on how each of them in the former formula can be mapped onto another in the latter. Differently from the case of objects, what defines the structure of a formula is the set of n-ary predicates, and specifically the way in which they are applied to the various objects to relate them (a predicate, applied to a number of terms equal to its arity, is called an *atom*). This is the most difficult part, since relations are specific to the first-order setting and are the cause of indeterminacy in mapping (parts of) a formula into (parts of) another one. In the following, we will call *compatible* two FOL (sub-)formulæ that can be mapped onto each other without yielding inconsistent term associations (i.e., a term in one formula cannot correspond to different terms in the other formula).

Table 1. Object Similarity

C			E		
t'	P'	R'	t''	P''	R''
X	$\{\pi,\phi,\rho\}$	$\{p/2.1,p/2.1,p/2.2\}$	a	$\{\pi,\phi,\sigma,\tau\}$	$\{p/2.1,p/2.1,p/2.2\}$
Y	$\{\pi,\sigma,\tau\}$	$\{p/2.2,r/2.1,o/2.1\}$	b	$\{\sigma,\tau\}$	$\{p/2.2,r/2.1,o/2.1\}$
Z	$\{\phi\}$	$\{p/2.2,o/2.2\}$	c	$\{\phi\}$	$\{p/2.2,o/2.2\}$
W	$\{\sigma,\tau\}$	$\{p/2.1,p/2.1,p/2.2\}$	d	$\{\tau,\rho\}$	$\{p/2.1,p/2.1\}$
U	$\{\pi,\phi\}$	$\{r/2.2,s/2.1\}$	f	$\{\pi,\phi,\sigma\}$	$\{r/2.2,t/2.1\}$

t'/t''	$(P'\setminus P''),(P'\cap P''),(P''\setminus P')$		$(R'\setminus R''),(R'\cap R''),(R''\setminus R')$		$\mathrm{sf}_o(t',t'')$
X/a	$\{\rho\},\{\pi,\phi\},\{\sigma,\tau\}$	$(1,2,2)$	$\emptyset,\{p/2.1,p/2.1,p/2.2\},\emptyset$	$(0,3,0)$	1.35
Y/b	$\{\pi\},\{\sigma,\tau\},\emptyset$	$(1,2,0)$	$\emptyset,\{p/2.2,r/2.1,o/2.1\},\emptyset$	$(0,4,0)$	1.46
Y/c	$\{\pi,\sigma,\tau\},\emptyset,\{\phi\}$	$(3,0,1)$	$\{r/2.1,o/2.1\},\{p/2.2\},\{o/2.2\}$	$(2,1,1)$	0.72
Z/b	$\{\phi\},\emptyset,\{\sigma,\tau\}$	$(1,0,2)$	$\{o/2.2\},\{p/2.2\},\{r/2.1,o/2.1\}$	$(1,1,2)$	0.74
Z/c	$\emptyset,\{\phi\},\emptyset$	$(0,1,0)$	$\emptyset,\{p/2.2,o/2.2\},\emptyset$	$(0,2,0)$	1.42
W/d	$\{\sigma\},\{\tau\},\{\rho\}$	$(1,1,1)$	$\{p/2.2\},\{p/2.1,p/2.1\},\emptyset$	$(1,2,0)$	1.18
U/f	$\emptyset,\{\pi,\phi\},\{\sigma\}$	$(0,2,1)$	$\{s/2.1\},\{r/2.2\},\{t/2.1\}$	$(1,1,1)$	1.18

Given an n-ary literal, we define its *star* as the multiset of n-ary predicates corresponding to the literals linked to it by some common term (indeed, a predicate can appear in multiple instantiations among these literals). Intuitively, the star of a literal depicts 'in breadth' how it relates to the rest of the formula. The *star similarity* $\mathrm{sf}_s(l',l'')$ between two compatible n-ary literals l' and l'' having stars S' and S'', respectively, can be computed as $\mathrm{sf}(n_s,l_s,m_s)$ for the following parameters:

$n_s = |S' \setminus S''|$ expresses how many more relations l' has in C' than l'' has in C'' (*star residual* of l' wrt l'');
$l_s = |S' \cap S''|$ is the number of relations that both l' in C' and l'' in C'' have in common;
$m_s = |S'' \setminus S'|$ expresses how many more relations l'' has in C'' than l' has in C' (*star residual* of l'' wrt l').

Overall, a more adequate evaluation of similarity between l' and l'' can be obtained by adding to the above result the characteristic and relational similarity values for all pair of their arguments in corresponding positions:

$$\mathrm{sf}_s(l',l'') = \mathrm{sf}(n_s,l_s,m_s) + \Sigma_{t'/t''\in\theta}\mathrm{sf}_o(t',t'')$$

where θ is the set of term associations that map l' onto l''.

Then, any first-order logic formula can be represented as a graph in which atoms are the nodes, and edges connect two nodes *iff* they are related in some way. It follows that a comparison between two formulæ to assess their structural similarity corresponds to the computation of (sub-)graph omomorphisms, a problem known to be *NP*-hard due to the possibility of mapping a (sub-)graph onto another in many different ways. As a consequence, we are interested in heuristics that can give significant hints on the structure overlapping between two formulæ

Algorithm 1 Construction of the graph associated to C

```
Require: $C = l_0 : -l_1, \dots, l_n$: Clause
  $i \leftarrow 0$; $Level_0 \leftarrow \{l_0\}$; $E \leftarrow \emptyset$; $Atoms \leftarrow \{l_1, \dots, l_n\}$
  while $Atoms \neq \emptyset$ do
    $i \leftarrow i + 1$
    $Level_i \leftarrow \{l \in Atoms \mid \exists l' \in Level_{i-1} s.t. terms(l) \cap terms(l') \neq \emptyset\}$
    $E \leftarrow E \cup \{(l', l'') \mid l' \in Level_{i-1}, l'' \in Level_i, terms(l') \cap terms(l'') \neq \emptyset\}$
    $Atoms \leftarrow Atoms \setminus Level_i$
  end while
  return $G = (\bigcup_i Level_i, E)$: graph associated to $C$
```

with little computational effort. The graph representation is obviously easier for clauses than for general formulæ, since they are made up by just a single atom in the head and a conjunction of atoms in the body. In the following, we will deal with *linked* clauses only (i.e. clauses whose associated graph is connected), and will build the graph based on a simple (as to the details it expresses about the fomula), yet powerful (as regards the information it conveys) feature, that is term sharing between couples of atoms. Given a clause C, we define its *associated graph* as $G_C = (V, E)$ with

– $V = \{l_0\} \cup \{l_i | i \in \{1, \dots, n\}, l_i$ built on k-ary predicate, $k > 1\}$ and
– $E \subseteq \{(a_1, a_2) \in V \times V \mid terms(a_1) \cap terms(a_2) \neq \emptyset\}$

where $terms(a)$ denotes the set of terms that appear as arguments of atom a. The strategy for choosing the edges to be represented, summarized in Algorithm 1, leverages on the presence of a single atom in the head to have both a starting point and precise directions for traversing the graph in order to choose a unique and well-defined perspective on the clause structure among the many possible. More precisely, we build a Directed Acyclic Graph (DAG), *stratified* (i.e., with the set of nodes partitioned) in such a way that the head is the only node at level 0 (first element of the partition) and each successive level (element of the partition) is made up by new nodes (not yet reached by edges) that have at least one term in common with nodes in the previous level. In particular, each node in the new level is linked by an incoming edge to each node in the previous level having among its arguments at least one term in common with it.

Example 3. Let us build the graph G_C. The head represents the 0-level of the stratification. Then directed edges may be introduced from $h(X)$ to $p(X, Y)$, $p(X, Z)$ and $p(W, X)$, that are the only atoms having X as an argument, which yields level 1 of the term stratification. Now the next level can be built, adding directed edges from atoms in level 1 to the atoms not yet considered that share a variable with them: $r(Y, U)$ – end of an edge starting from $p(X, Y)$ –, $o(Y, Z)$ – end of edges starting from $p(X, Y)$ and $p(X, Z)$ – and $q(W, W)$ – end of an edge starting from $p(W, X)$. The third and last level of the graph includes the only remaining atom, $s(U, V)$ – having an incoming edge from $r(Y, U)$.

Table 2. Star Similarity

C		E	
l'	S'	l''	S''
$p(X,Y)$	$\{p/2, p/2, r/2, o/2\}$	$p(a,b)$	$\{p/2, p/2, r/2, o/2\}$
$p(X,Z)$	$\{p/2, p/2, o/2\}$	$p(a,c)$	$\{p/2, p/2, o/2\}$
$r(Y,U)$	$\{p/2, o/2, s/2\}$	$r(b,f)$	$\{p/2, o/2, t/2\}$

l'	l''	$(S' \setminus S''), (S' \cap S''), (S'' \setminus S')$	$\mathrm{sf}_s(l', l'')$
$p(X,Y)$	$p(a,b)$	$\emptyset, \{p/2, p/2, r/2, o/2\}, \emptyset$ $(0,4,0)$	3.52
$p(X,Y)$	$p(a,c)$	$\{r/2\}, \{p/2, p/2, o/2\}, \emptyset$ $(1,3,0)$	2.80
$p(X,Z)$	$p(a,c)$	$\emptyset, \{p/2, p/2, o/2\}, \emptyset$ $(0,3,0)$	3.57
$p(X,Z)$	$p(a,b)$	$\emptyset, \{p/2, p/2, o/2\}, \{r/2\}$ $(0,3,1)$	2.82
$r(Y,U)$	$r(b,f)$	$\{s/2\}, \{p/2, o/2\}, \{t/2\}$ $(1,2,1)$	3.24

Now, all possible paths starting from the head and reaching *leaf* nodes (those with no outcoming edges) can be interpreted as the basic components of the overall structure of the clause. Being such paths univoquely determined reduces the amount of indeterminacy in the comparison. Intuitively, a path depicts 'in depth' a portion of the relations described in the clause. Given two clauses C' and C'', we define the *intersection* between two paths $p' =< l'_1, \ldots, l'_{n'} >$ in $G_{C'}$ and $p'' =< l''_1, \ldots, l''_{n''} >$ in $G_{C''}$ as the pair of longest compatible initial subsequences of p' and p'':

$p' \cap p'' = (p_1, p_2) = (< l'_1, \ldots, l'_k >, < l''_1, \ldots, l''_k >)$ s.t.
$\forall i = 1, \ldots, k : l'_1, \ldots, l'_i$ compatible with $l''_1, \ldots, l''_i \wedge$
$(k = n' \vee k = n'' \vee l'_1, \ldots, l'_{k+1}$ incompatible with $l''_1, \ldots, l''_{k+1})$
and the two residuals as the incompatible trailing parts:

$p' \setminus p'' =< l'_{k+1}, \ldots, l'_{n'} >, p'' \setminus p' =< l''_{k+1}, \ldots, l''_{n''} >)$

Hence, the *path similarity* between p' and p'', $\mathrm{sf}_s(p', p'')$, can be computed by applying (1) to the following parameters:

$n_p = |p' \setminus p''| = n' - k$ is the length of the trail incompatible sequence of p' wrt p'' (*path residual* of p' wrt p'');

$l_p = |p_1| = |p_2| = k$ is the length of the maximum compatible initial sequence of p' and p'';

$m_p = |p'' \setminus p'| = n'' - k$ is the length of the trail incompatible sequence of p'' wrt p' (*path residual* of p'' wrt p').

plus the star similarity of all couples of literals in the initial sequences:

$$\mathrm{sf}_p(p', p'') = \mathrm{sf}(n_p, l_p, m_p) + \Sigma_{i=1,\ldots,k} \mathrm{sf}_s(l'_i, l''_i)$$

Example 4. Since the head is unique (and hence can be uniquely matched), in the following we will deal only with the body literals for structural criteria. Table 2 reports the star comparisons for a sample of literals in C and E, while Table 3 shows some path comparisons.

Table 3. Path Similarity

Path No.	C	E
1.	$< p(X,Y), r(Y,U), s(U,V) >$	$< p(a,b), r(b,f), t(f,g) >$
2.	$< p(X,Y), o(Y,Z) >$	$< p(a,b), o(b,c) >$
3.	$< p(X,Z), o(Y,Z) >$	$< p(a,c), o(b,c) >$
4.	$< p(W,X), q(W,W) >$	$< p(d,a), q(d,e) >$

p'	$p' \cap p''$	$p' \setminus p''$	$\theta_{p' \cap p''}$	$(n,l,m)_p$
p''		$p'' \setminus p'$		$\mathrm{sf}_p(p',p'')$
C.1	$< p(X,Y), r(Y,U) >$	$< s(U,V) >$	$\{X/a, Y/b, U/f\}$	$(1,2,1)$
E.1	$< p(a,b), r(b,f) >$	$< t(f,g) >$		7.36
C.1	$< p(X,Y) >$	$< r(Y,U), s(U,V) >$	$\{X/a, Y/b\}$	$(2,1,1)$
E.2	$< p(a,b) >$	$< o(b,c) >$		3.97
C.2	$< p(X,Y) >$	$< o(Y,Z) >$	$\{X/a, Y/b\}$	$(1,1,2)$
E.1	$< p(a,b) >$	$< r(b,f), t(f,g) >$		3.97
C.2	$< p(X,Y), o(Y,Z) >$	$<>$	$\{X/a, Y/b, Z/c\}$	$(0,2,0)$
E.2	$< p(a,b), o(b,c) >$	$<>$		7.95

Note that no single criterion is by itself neatly discriminant, but their cooperation succeeds in distributing the similarity values and in making the difference ever clearer as long as they are composed one ontop the previous ones.

Now, a generalization can be computed considering the path intersections by decreasing similarity, adding to the partial generalization generated thus far the common literals of each pair whenever they are compatible (see Algorithm 2). Further generalizations can then be obtained through backtracking. This optionally allows to cut the generalization when some length threshold is reached, ensuring that only the less significant similarities are dropped.

Example 5. The path intersection with highest similarity value is C.3/E.3, and hence the first partial generalization becomes $\{p(X,Z), o(Y,Z)\}$, with associations $\{X/a, Y/b, Z/c\}$. Then C.2/E.2 is considered, whose associations are compatible with the current ones, so it contributes with $\{p(X,Y)\}$ to the generalization (there are no new associations). Then comes C.1/E.1, that being compatible extends the generalization by adding $\{r(Y,U)\}$ and the association with $\{U/f\}$. It is the turn of C.1/E.2 and then of C.2/E.1, that are compatible but redundand, and hence do not add anything to the current generalization (nor to the associations). Then C.4/E.4 is considered, that is compatible and extends with $\{p(W,X)\}$ and $\{W/d\}$ the current generalization and associations, respectively. Lastly C.3/E.2, C.2/E.3, C.3/E.1 and C.1/E.3 are considered, but discarded because of their associations being incompatible.

4 Related Works

Despite of many distance measures developed for attribute-value representations, few works faced the definition of similarity or distance measures for first-order descriptions. In [4] a distance measure is proposed, based on probability theory

Algorithm 2 Similarity-based generalization

```
Require: $C$: Rule; $E$: Example
  $P_C \leftarrow paths(C)$; $P_E \leftarrow paths(E)$;
  $P \leftarrow \{(p_C, p_E) \in P_C \times P_E | p_C \cap p_E \neq (<>, <>)\}$;
  $G \leftarrow \emptyset$; $\theta \leftarrow \emptyset$
  while $P \neq \emptyset$ do
    $(\overline{p}_C, \overline{p}_E) \leftarrow \text{argmax}_{(p_C, p_E) \in P}(sf(p_C, p_E))$
    $P \leftarrow P \setminus \{(\overline{p}_C, \overline{p}_E)\}$
    $(\overline{q}_C, \overline{q}_E) \leftarrow p_C \cap p_E$
    $\theta_q \leftarrow$ substitution s.t. $q_C = q_E$
    if $\theta_q$ compatible with $\theta$ then
      $G \leftarrow G \cup q_C$; $\theta \leftarrow \theta \cup \theta_q$
    end if
  end while
  return $G$: generalization between $C$ and $E$
```

applied to the formula components. Compared to that, our function does not require the assumptions and simplifying hypotheses (statistical independence, mutual exclusion) to ease the probability handling, and no *a-priori* knowledge of the representation language is required (such as the type domains). It does not require the user to set weights on the predicates' importance, and is not based on the presence of 'mandatory' relations, like for the $G1$ subclause in [4].

Many supervised learning systems prove the importance of a distance measure. For instance, *KGB* [1] uses a similarity function, parameterized by the user, to guide generalization; our ideas of characteristic and relational similarity are very close to those, but the similarity computation is more straightforward. While *KGB* cannot handle negative information in the clauses, our approach can be easily extended to do that. The k-Nearest Neighbor classifier *RIBL* [2] is based on a modified version of the function proposed in [1]. The basic idea is that object similarity depends on the similarity of their attributes' values (the similarity of their size) and, recursively, on the similarity of the objects related to them. Such a propagation poses the problem of indeterminacy in associations, that our technique avoids thanks to the different structural approach.

[9] presents an approach for the induction of a distance on FOL examples, that depends on the pattern discriminating the target concepts. k clauses are choosen and the truth values of whether each clause covers the example or not are used as k features for a distance on the space $\{0, 1\}^k$ between the examples. [6] organizes terms in an importance-related hierarchy, and proposes a distance between terms based on interpretations and a level mapping function that maps every simple expression on a natural number. [7] presents a distance function between atoms based on the difference with their lgg, and uses it to compute distances between clauses. It consists of a pair: the first component extends the distance in [6] and is based on the differences between the functors on both terms, while the second component is based on the differences in occurrences of variables and allows to differentiate cases where the first component cannot.

Table 4. Experimental results

		Ratio	Time (sec.)	Cl	Gen	Exc^+	$Spec^+$	$Spec^-$	Exc^-	Acc
Classification	SF	90.52	579	8	47(+0)	0	2	0	0	0.94
	I	70.22	137	7	33(+100)	0	1	1	1	0.97
	S80	73.63	206	7	33(+13)	0	0	1	1	0.97
Labelling	SF	91.09	22220	36	180(+0)	0	8	3	3	0.89
	I	68.85	33060	39	137(+5808)	0	15	11	12	0.93
	S80	71.75	15941	54	172(+220)	0	14	8	2	0.93

5 Experiments

To check whether the proposed criteria are actually able to give significant similarity hints when comparing two structures, the similarity-driven generalization procedure was compared to a previous non-guided procedure, embedded in the learning system INTHELEX [3]. The system was set so that, whenever the first generalization returned by the guided procedure was not consistent with all past negative examples, the system could search for more specific ones on backtracking[1]. 10-fold cross-validation was exploited to assess predictive accuracy.

A first comparison concerned the classical ILP Mutagenesis dataset, where it reached a slightly better predictive accuracy (87%) than the non-guided version (86%) exploiting only 30% runtime (4035 seconds instead of 13720). Noteworthy, the first generalization found was always correct, and thus no backtracking was ever required to search for other alternatives, whereas the non-guided version computed 5815 additional generalizations to takle cases in which the first generalization found was inconsistent with past examples. This confirmed that the similarity criteria, strategy and formula are able to lead the correct identification of corresponding sub-parts of the compounds descriptions, and convinced us to investigate more deeply on the system behaviour. Other experiments were run on a dataset[2] containing 122 descriptions of scientific papers first page layout, belonging to 4 different classes and corresponding to 488 positive/negative examples for learning document classification rules plus 1488 examples for learning rules to identify 12 kinds of significant logical roles document components (e.g., title, author, abstract). Results are reported in Table 4.

The first question concerns whether the proposed similarity function is actually able to lead towards the identification of the proper sub-parts to be put in correspondence in the two descriptions under comparison. Since the 'correct' association is not known, this can be evaluated only indirectly. A way for doing this is evaluating the 'compression' factor of the guided generalization, i.e., the portion of literals in the clauses to be generalized that is preserved by the generalization. Indeed, since each generalization in INTHELEX must be a subset

[1] In order to avoid the system to waste too much time on difficult generalizations, it was set so to give up and try to generalize another clause (if any) whenever 500 redundant (sub-)generalizations were found on backtracking, or 50 new iconsistent ones were computed, which happened first.

[2] `http://lacam.di.uniba.it:8000/systems/inthelex/index.htm#datasets`

of either clause to be generalized (because of the Object Identity assumption), the more literals the generalization preserves from these clause, the less general it is. More formally, we evaluate the compression as the ratio between the length of the generalization and that of the shortest clause to be generalized: the higher such a value, the more confident one can be that the correct associations were provided by the similarity criteria and formula. Of course, the more the difference in length between the two clauses to be generalized, the more indeterminacy is present, and hence the more difficult it is to identify the proper corresponding parts between them. Interestingly, on the document dataset the similarity-driven generalization (SF) preserved on average more than 90% literals of the shortest clause, with a maximum of 99,48% (193 literals out of 194, against an example of 247) and just 0,006 variance. As a consequence, one woud expect that the produced generalizations are least general ones or nearly so. To check this, instead of computing the actual least general generalization for each performed generalization and compare it to that returned by the guided procedure, that would have been computationally heavy and would require modifying the system behaviour, we counted how many backtrackings the system had to perform in order to reach a more specific one. The outcome was that no more specific generalization was ever found within the given limit, which suggests that the first generalization found is likely to be very near to (and indeed it often is just) the least general one. Note that application of Tverski's similarity formula [10] (a state-of-the-art one in the current literature) to the same setting always returned shortest generalizations than those obtained by using our formula.

However, being such generalizations very specific, they show lower predictive accuracy than the non-guided INTHELEX algorithm (I), probably due to the need of more examples for converging to more predictive definitions or to overfitting. For this reason, the similarity-driven generalization was bound to discard at least 20% literals of the shortest original clause (S80): this led to the same predictive accuracy as I, and dramatically reduced runtime with respect to the unbound version (and also to I on the labelling task). The number of generalizations also decreases, although at the cost of more specialization effort (also by means of negative literals, that are handled as suggested in the previous section), that is in any case more effective than in I (particularly as regards negative exceptions). In the classification task the number of clauses slightly decreases, while in the labelling task it increases of 15, but balanced by 10 negative exceptions less. Noteworthy, the generalizations found are still very tight to the examples, since in the classification experiment only 13 more specific generalizations (of which only 1 correct) were found (and tested for correctness) by backtracking, whereas I found 100, of which 6 correct. In the labelling task, $S80$ found 220 additional candidate generalizations (of which 6 correct) against 5808 (of which 39 correct) of I. This means that one could even avoid backtracking with small loss in $S80$, but not in I.

Another important parameter for assessing the usefulness of the proposed function and strategy is runtime. Table 4 reveals that, on the labelling task, using the similarity function leads to savings that range from 1/3 up to 1/2 of the time, in the order of hours, also on this dataset. The performance on the classification task is in any case comparable (difference in the order of tens of seconds), and can probably be explained with the fact that such a task is easier, so there is little space for improvement and the time needed for computing the formula nullifies the savings.

6 Conclusions

Relations in First-Order Logic lead to indeterminacy in mapping portions of a description onto another one. In this paper we identify a set of criteria and a formula for clause comparison, and exploit it to guide a generalization procedure by indicating the subparts of the descriptions that are more likely to correspond to each other, based only on their syntactic structure. According to the experimental outcomes, the similarity-based generalization is able to capture the correct associations, and can get the same predictive accuracy as the non-guided version, but yielding 'cleaner' theories and dramatically reducing the amount of time required to converge (interestingly, time savings increase as long as the problem complexity grows). Future work will concern fine-tuning of the similarity computation methodology, and its application to other problems, such as flexible matching, conceptual clustering and instance-based learning.

References

[1] G. Bisson. Learning in FOL with a similarity measure. In W.R. Swartout, editor, *Proc. of AAAI-92*, pages 82–87, 1992.

[2] W. Emde and D. Wettschereck. Relational instance based learning. In L. Saitta, editor, *Proc. of ICML-96*, pages 122–130, 1996.

[3] F. Esposito, S. Ferilli, N. Fanizzi, T. Basile, and N. Di Mauro. Incremental multistrategy learning for document processing. *Applied Artificial Intelligence Journal*, 17(8/9):859–883, 2003.

[4] F. Esposito, D. Malerba, and G. Semeraro. Classification in noisy environments using a distance measure between structural symbolic descriptions. *IEEE Transactions on PAMI*, 14(3):390–402, 1992.

[5] Dekang Lin. An information-theoretic definition of similarity. In *Proc. 15th International Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.

[6] S. Nienhuys-Cheng. Distances and limits on herbrand interpretations. In D. Page, editor, *Proc. of ILP-98*, volume 1446 of *LNAI*, pages 250–260. Springer, 1998.

[7] J. Ramon. *Clustering and instance based learning in first order logic.* PhD thesis, Dept. of Computer Science, K.U.Leuven, Belgium, 2002.

[8] C. Rouveirol. Extensions of inversion of resolution applied to theory completion. In *Inductive Logic Programming*, pages 64–90. Academic Press, 1992.

[9] M. Sebag. Distance induction in first order logic. In N. Lavrač and S. Džeroski, editors, *Proc. of ILP-97*, volume 1297 of *LNAI*, pages 264–272. Springer, 1997.

[10] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.